FUJITSU

FUJITSU Software
SIMPLIA TF-MDPORT Pro (64bit) V81L20

ユーザーズガイド　API編

Windows(64)

SIMPLIA-MDPWI64-06-JP81(03)
2024年3月

まえがき

SIMPLIA TF-MDPORT Proは、ソフトウェア開発保守支援システムの一つとして位置づけられ、汎用機・オフコン・UNIX機・PCとの間で、データやソースの流通を支援するツールです。

本書は、SIMPLIA TF-MDPORT ProのAPIについて記述しています。

本書の目的

SIMPLIA TF-MDPORT ProのAPIについて理解できることを目的としています。

本書の読者

SIMPLIA TF-MDPORT ProのAPIを使用される方を対象としています。

本書を読むためには、以下の知識が必要です。

- 文字コードに関する基本的な知識 (シフトJISコード、EUCコード、汎用機コード、Unicode等)
- ご使用になるOSに関する基本的な知識

本書の構成

以下の構成で記述しています。

- 第1章 概要

  SIMPLIA TF-MDPORT ProのAPIの概要について説明します。

- 第2章 機能説明

  SIMPLIA TF-MDPORT ProのAPIの機能について説明します。

- 第3章 注意事項

  SIMPLIA TF-MDPORT ProのAPIの注意事項について説明します。

- 付録A 関数一覧

  SIMPLIA TF-MDPORT ProのAPIの各関数の仕様、インターフェースについて説明します。

- 付録B 使用例

  SIMPLIA TF-MDPORT ProのAPIの使用例を説明します。

本書の表記について

本書では、説明するうえで、次の略記を使用しています。

「Windows(R) 11 Home」、 「Windows(R) 11 Pro」、 「Windows(R) 11 Enterprise」、または 「Windows(R) 11 Education」	→	「Windows 11」
「Windows(R) 10 Home」、 「Windows(R) 10 Pro」、 「Windows(R) 10 Enterprise」、または 「Windows(R) 10 Education」	→	「Windows 10」

「Microsoft(R) Windows Server(R) 2022 Datacenter」、 「Microsoft(R) Windows Server(R) 2022 Standard」、または 「Microsoft(R) Windows Server(R) 2022 Essentials」	→	「Windows Server 2022」
「Microsoft(R) Windows Server(R) 2019 Datacenter」、 「Microsoft(R) Windows Server(R) 2019 Standard」、または 「Microsoft(R) Windows Server(R) 2019 Essentials」	→	「Windows Server 2019」
「Microsoft(R) Windows Server(R) 2016 Datacenter」、 「Microsoft(R) Windows Server(R) 2016 Standard」、または 「Microsoft(R) Windows Server(R) 2016 Essentials」	→	「Windows Server 2016」
次の製品すべてを指す場合 Windows 11 Windows 10 Windows Server 2022 Windows Server 2019 Windows Server 2016	→	「Windows」
「Interstage Charset Manager Standard Edition Agent」	→	「Charset Manager」
「SIMPLIA TF-MDPORT Pro」、または 「SIMPLIA TF-MDPORT Pro (64bit)」	→	「MDPORT」

本書では、次のような表記法を用います。

- 16進数には、“0xXX”のような表記を用います。ここで、“X”は16進数字です。桁数は、必要に応じて、2～8桁になります。ただし、16進数で表記していることが文脈から明らかな場合には、“0x”を省略し、単に“XX”のように表記することがあります。
- 値の範囲には、“X～X”のような表記を用います。範囲には、両端の値を含みます。

本書では、C言語の形式で説明しています。

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

登録商標について

本書で使われている登録商標および商標は、以下のとおりです。

- Microsoft、Windows、Windows Server、Microsoft Edge、その他のマイクロソフト製品の名称および製品名は、米国 Microsoft Corporation の、米国およびその他の国における登録商標または商標です。
- UNIXは、米国およびその他の国におけるオープン・グループの登録商標です。
- OracleとJavaは、Oracle Corporationおよびその子会社、関連会社の米国およびその他の国における登録商標です。
- Excelは、米国Microsoft Corporationの製品です。
- そのほか、本書に記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。

なお、本文中では、™マーク、®マークは省略しています。

2024年3月

目　次

第1章 概要 1

第2章 機能説明 2
2.1 機能一覧 2
2.2 レコード変換機能 2
2.2.1 機能説明 2
2.2.2 システム構成 2
2.2.3 関数一覧 4
2.2.4 使用方法 4
2.2.5 コンパイル方法 6
2.2.6 レコード変換機能(旧形式)からの移行のポイント 6
2.2.6.1 リンクするライブラリについて 6
2.2.6.2 レコード変換機能の各関数について 6
2.2.6.3 変換指示構造体について 7
2.2.6.4 変換エラーファイルの出力について 10

第3章 注意事項 11

付録A 関数一覧 13
A.1 MDP_init() 13
A.1.1 MDPcvinf構造体 14
A.2 MDP_conv() 22
A.3 MDP_fin() 23
A.4 MDP_stat() 23
A.5 MDP_errout() 24

付録B 使用例 26

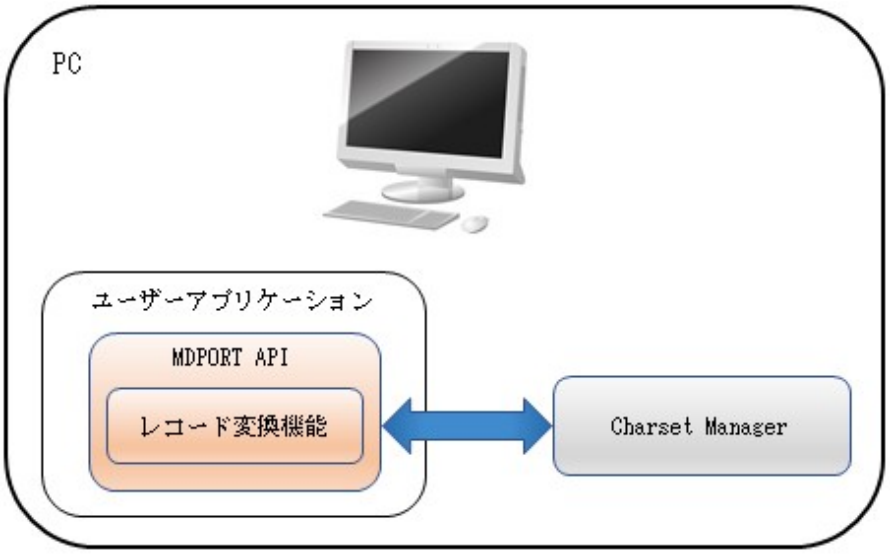
用語集 29

第1章 概要

MDPORTが提供するインターフェースをユーザーアプリケーションに組み込むことにより、文字コード・レコード形式変換を行うことができます。

C言語とCOBOLでのインターフェースを提供しています。

図1.1 MDPORT APIの概要図



注意

- 本バージョンには、旧形式のAPI(コード変換機能/レコード変換機能)は含まれていません。
- 本書では、旧バージョンで提供していたレコード変換機能を「レコード変換機能(旧形式)」と表記します。

参照

本書は、APIの内容に絞って説明しています。

MDPORTの特長、適用のケース、変換仕様、利用者定義変換テーブルの書式等については、画面操作と共通した内容のため、「ユーザーズガイド 画面操作編」を併せてお読みください。

第2章 機能説明

MDPORTのAPIの機能について説明します。

2.1 機能一覧

MDPORTのAPIには、以下の機能があります。

表2.1 機能一覧

機能	概要
レコード変換機能	レコード単位に、文字コードおよびレコード形式の変換を行うことができます。

2.2 レコード変換機能

レコード変換機能について説明します。

2.2.1 機能説明

ダイナミックリンクライブラリ(DLL)として提供する関数群を、ユーザーアプリケーションに組み込んで使用します。

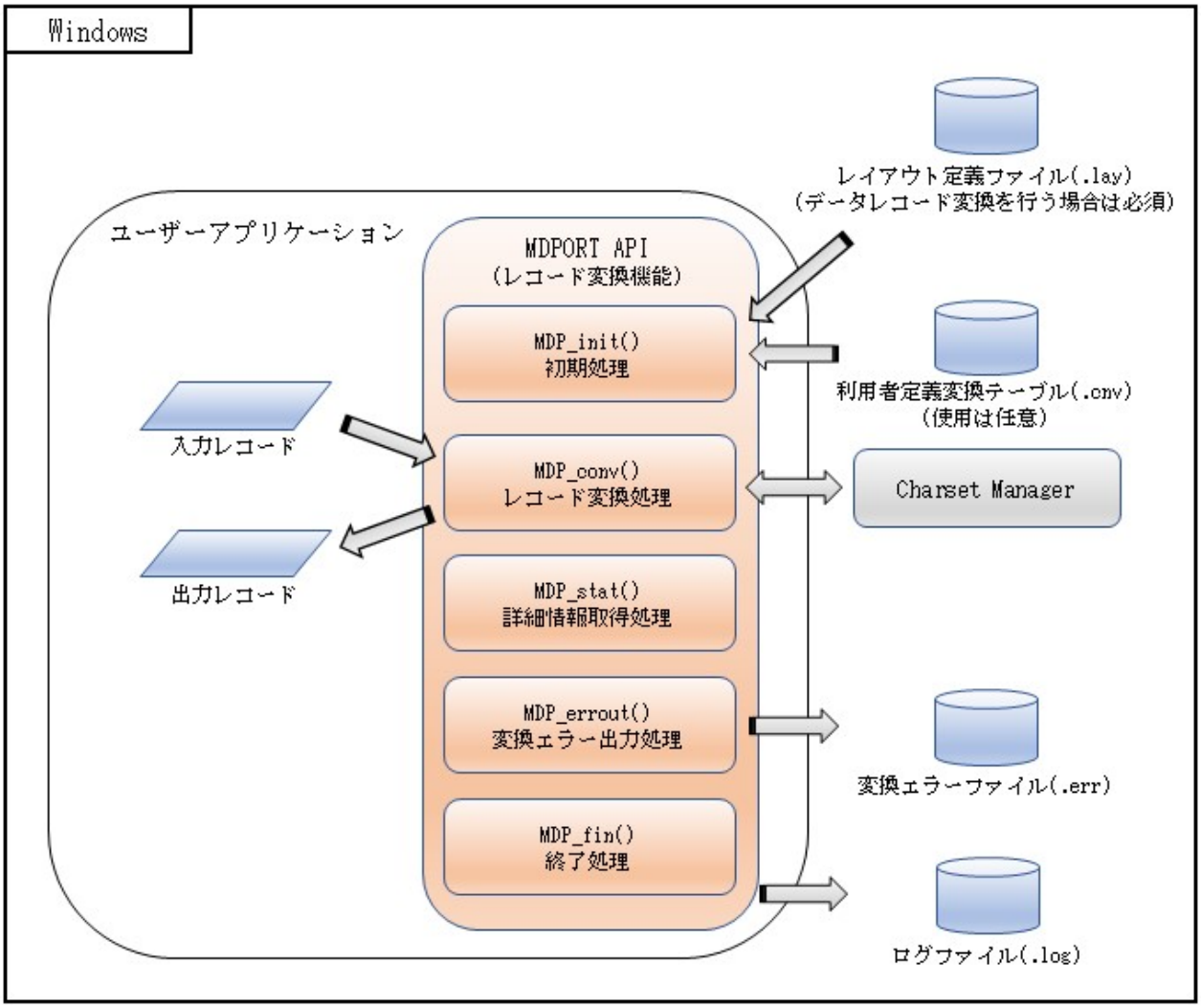
ユーザーアプリケーションからMDPORTに渡したレコードについて、文字コードおよびレコード形式の変換を行い、ユーザーアプリケーションへ返却します。

変換の指示内容は、ユーザーアプリケーションから指定します。

2.2.2 システム構成

レコード変換機能のシステム構成図を以下に示します。

図2.1 システム構成図



ポイント

- データレコード変換を行う場合、レイアウト定義ファイルを指定してレコードのフォーマット情報を指定します。
- 利用者定義変換テーブルの指定により、利用者定義文字(外字)や拡張文字の変換仕様を任意に指定することができます。
- システム間の文字コードの管理をCharset Managerで実現している場合、Charset Manager(標準コード変換)を使用した変換が行えます。
- 変換中に発生したエラーを、変換エラーファイルに出力することができます。

  変換エラーファイルの出力先は、MDPcvinf構造体の「convErrFile」で指定します。
- 実行履歴は、ログファイルへ出力されます。

  ログファイルの出力先は、MDPcvinf構造体の「logFile」で指定します。

  レコード変換機能では、変換処理の開始終了メッセージは出力しないため、異常時のメッセージのみログファイルに出力されます。

  処理が正常終了した場合でも、常にログファイルは作成されます。

参照

- レイアウト定義ファイルは、レイアウト定義機能(画面操作またはMDPORTLCコマンド)を使用して作成します。

  レイアウト定義機能(画面操作)については、「ユーザーズガイド 画面操作編」を参照してください。

  レイアウト定義のコマンド機能(MDPORTLCコマンド)については、「ユーザーズガイド コマンド編」を参照してください。
- 利用者定義変換テーブルについては、「ユーザーズガイド 画面操作編」の「付録A 利用者定義変換テーブル」を参照してください。
- 変換エラーファイルについては、「ユーザーズガイド 画面操作編」の「3.2.8 変換エラー表示」の「変換エラーファイルの出力内容」を参照してください。
- ログファイルについては、「ユーザーズガイド コマンド編」の「2.5.1 ログファイル出力」の「確認方法」を参照してください。

2.2.3 関数一覧

レコード変換機能で提供される関数は、以下のとおりです。

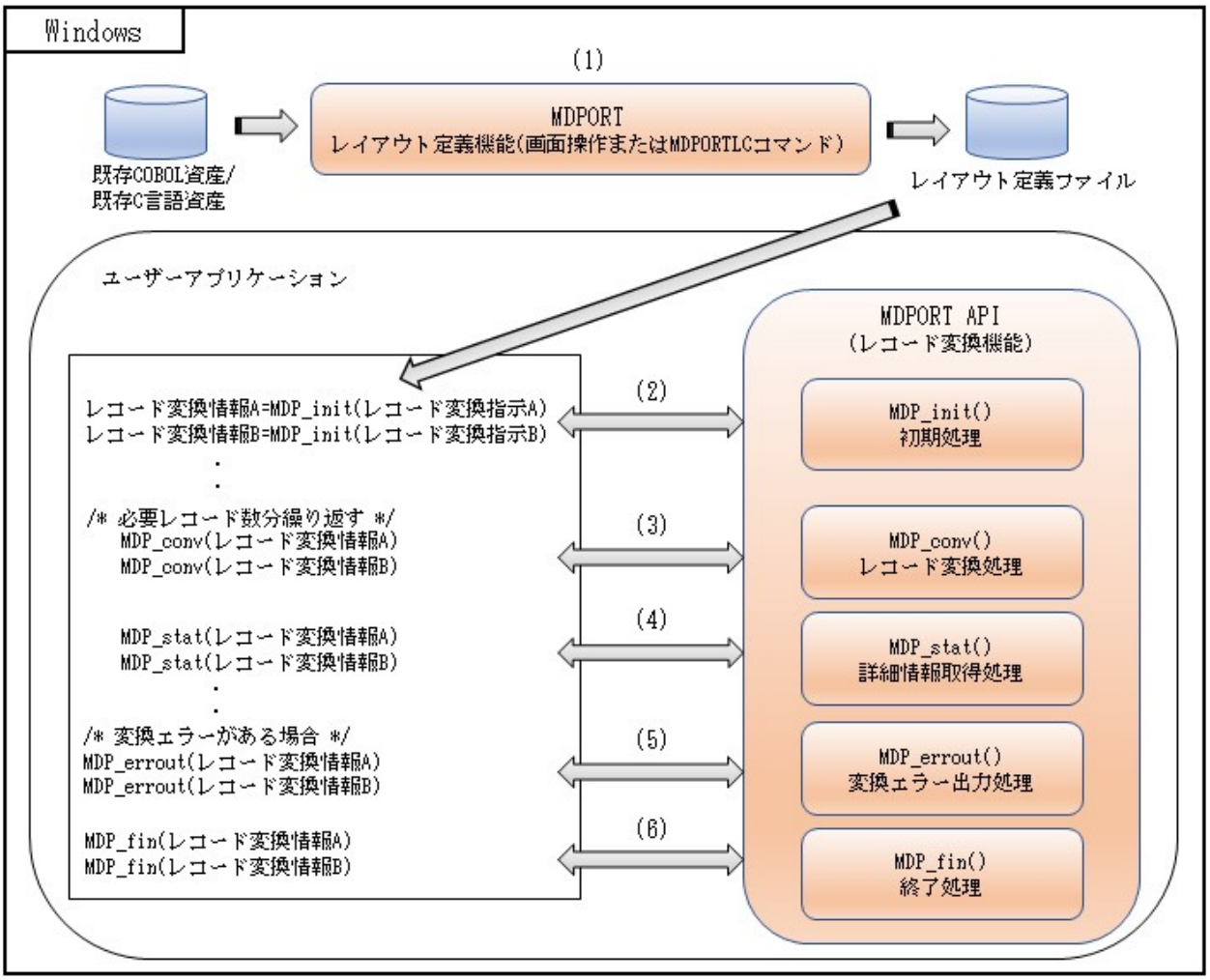
表2.2 関数一覧

No	関数名	機能
1	MDP_init()	変換パスの宣言、レイアウト定義ファイルの指定、変換に使用する領域の動的確保等、レコード変換機能の初期処理を行います。
2	MDP_conv()	MDP_init()関数で指示された内容に従って、1レコード分の文字コード・レコード形式変換を行います。
3	MDP_fin()	MDP_init()関数で確保した領域の解放等、レコード変換機能の終了処理を行います。
4	MDP_stat()	直前のMDP_conv()関数での詳細復帰情報を取得します。
5	MDP_errout()	レコード変換時に発生したエラーを収集し、変換エラーファイルへ出力します。

2.2.4 使用方法

レコード変換機能の使用方法は、以下のとおりです。

図2.2 使用イメージ



例) 1つのアプリケーションで複数の変換指示を指定する場合(レイアウト定義ファイルが異なる等)

表2.3 手順

No	内容
(1)	レイアウト定義機能(画面操作またはMDPORTLCコマンド)を使用して、レイアウト定義ファイルを作成します。 レイアウト定義ファイルは、データレコード変換を行う場合のみ必要です。テキスト変換の場合、この手順は不要です。
(2)	初期処理(MDP_init)により、レコード変換指示(変換パス、レイアウト定義ファイル名等)を宣言し、レコード変換情報の領域を確保します。 レコード変換処理(MDP_conv)を行う前に、一度だけ呼び出します。 レイアウト定義ファイルが異なる等、1つのアプリケーション内で複数のレコード変換指示を指定する場合は、その都度の初期処理(MDP_init)が必要です。
(3)	レコード変換処理(MDP_conv)により、1レコード分の文字コードの変換とレコード形式の変換を行います。 初期処理(MDP_init)と終了処理(MDP_fin)の間で、ユーザーアプリケーションで必要な回数(例えば、入力ファイルのレコード件数分)呼び出します。
(4)	詳細情報取得処理(MDP_stat)により、直前のレコード変換処理(MDP_conv)での詳細復帰値あるいはワーニング件数を取得します。 詳細情報取得処理(MDP_stat)は、必ずしも呼び出す必要はありません。
(5)	変換エラー出力処理(MDP_errout)により、変換エラー情報を変換エラーファイルに出力します。

No	内容
	すべてのレコード変換処理(MDP_conv)を終えた後に、一度だけ呼び出します。 変換エラー出力処理(MDP_errout)は、必ずしも呼び出す必要はありません。
(6)	終了処理(MDP_fin)により、初期処理(MDP_init)で確保したレコード変換情報の領域を解放し、終了処理を行います。 すべてのレコード変換処理(MDP_conv)を終えた後に、一度だけ呼び出します。

参照

レコード変換機能の具体的な組み込み方法については、「付録B 使用例」を参照してください。

2.2.5 コンパイル方法

レコード変換機能は、ダイナミックリンクライブラリ(DLL)として提供しています。

ユーザーアプリケーションのコンパイル時に、「F5AVPENG.lib」をリンクしてください。

2.2.6 レコード変換機能(旧形式)からの移行のポイント

レコード変換機能(旧形式)を使用している既存のアプリケーションを移行する場合の、ポイントについて説明します。

- リンクするライブラリについて
- レコード変換機能の各関数について
- 変換指示構造体について
- 変換エラーファイルの出力について

2.2.6.1 リンクするライブラリについて

コンパイル時にリンクするライブラリが、レコード変換機能(旧形式)から変更されました。

表2.4 コンパイル時にリンクするライブラリ

レコード変換機能(旧形式)	レコード変換機能
F5AVPREC.lib	F5AVPENG.lib

2.2.6.2 レコード変換機能の各関数について

以下の関数について、レコード変換機能(旧形式)から変更されました。

- 新規に追加された関数
- パラメタが変更された関数

- 復帰値が変更された関数

新規に追加された関数

変換エラー出力処理(MDP_errout)が新規に追加されました。

パラメタが変更された関数

初期処理(MDP_init)のパラメタの名称が変更されました。

表2.5 MDP_init()関数のパラメタ名称

レコード変換機能(旧形式)	レコード変換機能
inf	cvinf

復帰値が変更された関数

詳細情報取得処理(MDP_stat)について、“MDP_ERROR”を指定した場合の、復帰値の意味が変更されました。

表2.6 MDP_stat()関数の“MDP_ERROR”の場合の復帰値

レコード変換機能(旧形式)		レコード変換機能	
復帰値	意味	復帰値	意味
0	正常終了しました。	0	正常終了しました。
-1 (MDPERR_PARA)	引数に誤りがあります。	-1 (MDPERR_PARA)	引数に誤りがあります。
-2 (MDPERR_CONV)	コード変換エラー(ワーニング)が発生しました。	-2 (MDPERR_CONV)	コード変換エラー(ワーニング)が発生しました。
-3 (MDPERR_ENV)	環境に誤りがあります。	-3 (MDPERR_LLNG)	入出力レコード長に誤りがあります。
		-4 (MDPERR_ERROVER)	警告メッセージの件数が上限値を超えました。
		-5 (MDPERR_TRYUSE)	体験版の最大レコード数を超えました。
その他の数値	システムのエラーナンバーです。	その他の数値	システムのエラーナンバーです。

2.2.6.3 変換指示構造体について

変換に必要な情報を設定する変換指示構造体(MDPcvinf)について、メンバの名称がレコード変換機能(旧形式)から変更されました。

MDPcvinf構造体の詳細については、「付録A 関数一覧」の「A.1.1 MDPcvinf構造体」を参照してください。

その他、設定の変更点を以下に示します。

- メンバの一部削除
- Charset Manager iconv_open関数キーワードの指定方法の変更
- 「1バイト系代替コード」および「2バイト系代替コード」の型の変更
- 「メッセージ出力MAX件数」の指定値による動作の変更

- 「MSG上限到達時の処理」の省略時の動作の変更
- Charset Manager使用有無の指定方法の変更

メンバの一部削除

レコード変換機能(旧形式)の以下のMDPcvinf構造体メンバが、レコード変換機能では無くなりました。

表2.7 レコード変換機能での動作

レコード変換機能(旧形式)におけるMDPcvinf構造体メンバ	レコード変換機能での動作
itype (入力ファイル形式)	レイアウト定義ファイルの入力データ形式に従います。
otype (出力ファイル形式)	レイアウト定義ファイルの出力データ形式に従います。
isomode (ISO/エミュレータモード)	エミュレータモードで動作します。
ikana (入力半角カナモード)	EUCコード体系のコードセット2(2バイト/1文字)で動作します。
okana (出力半角カナモード)	EUCコード体系のコードセット2(2バイト/1文字)で動作します。
irecfm (入力レコード属性)	固定長で動作します。 (旧形式での動作と変更なし)
orecfm (出力レコード属性)	固定長で動作します。 (旧形式での動作と変更なし)
shift (シフトコード除去モード)	シフトコード除去後、左詰めで動作します。
cs3mode (コードセット3モード)	EUCコード体系のコードセット3(3バイト/1文字)有効で動作します。
noblc (後続空白削除防止フラグ)	後続空白削除で動作します。 (X項目:半角空白削除、N項目:全角空白削除)

Charset Manager iconv_open関数キーワードの指定方法の変更

Charset Manager変換でのiconv_open関数キーワードを、MDPcvinf構造体で指定するようになりました。

表2.8 iconv_open関数キーワードの指定方法

レコード変換機能(旧形式)	レコード変換機能
環境変数で指定します。	MDPcvinf構造体の「inCodeKey」および「outCodeKey」で指定します。

「1バイト系代替コード」および「2バイト系代替コード」の型の変更

MDPcvinf構造体メンバの「1バイト系代替コード」および「2バイト系代替コード」の型が、レコード変換機能(旧形式)から変更されました。

表2.9 代替コードを指定するMDPcvinf構造体メンバ

レコード変換機能(旧形式)		レコード変換機能	
MDPcvinf構造体メンバ	型	MDPcvinf構造体メンバ	型
amend1 (1バイト系代替コード)	int	outAlt1Char (1バイト系代替コード)	char
amend2 (2バイト系代替コード)	int	outAlt2Char (2バイト系代替コード)	char

「メッセージ出力MAX件数」の指定値による動作の変更

MDPcvinf構造体メンバの「メッセージ出力MAX件数」の指定値による動作が、レコード変換機能(旧形式)から変更されました。

表2.10 「メッセージ出力MAX件数」の指定値による動作

指定値	レコード変換機能(旧形式) メンバ名：msgmax	レコード変換機能 メンバ名：errMsgMax
-1	変換エラーを出力しません。	“0～9999”以外の指定値は不正な値とみなし、異常終了します。 旧形式と同様に、変換エラーを出力したくない場合は、変換エラー出力処理(MDP_errout)を呼び出さないようにしてください。
0	無制限に変換エラーを出力します。	デフォルト値“100”が指定されたとみなします。
1～9999	指定した件数まで変換エラーを出力します。	レコード変換機能(旧形式)と動作に変更はありません。

「MSG上限到達時の処理」の省略時の動作の変更

MDPcvinf構造体のメンバの「MSG上限到達時の処理」の省略値が、レコード変換機能(旧形式)から変更されました。

表2.11 「MSG上限到達時の処理」の省略値

レコード変換機能(旧形式) メンバ名：msgcont	レコード変換機能 メンバ名：errMsgNonStop
“MSGCONT_ON”として扱い、処理を続行します。	“ERRMSGNONSTOP_STOP”として扱い、処理を中止します。

Charset Manager使用有無の指定方法の変更

Charset Managerの使用有無を、MDPcvinf構造体の「iconvUse」で指定するようになりました。

表2.12 Charset Managerの使用有無の指定

レコード変換機能(旧形式)	レコード変換機能
Charset Managerの使用有無の指定はできません。 Charset Managerがインストールされている場合は、Charset Managerを使用します。	MDPcvinf構造体の「iconvUse」で、Charset Managerの使用有無を指定します。

2.2.6.4 変換エラーファイルの出力について

レコード変換処理(MDP_conv)で発生した変換エラーについて、変換エラーファイルの出力方法が、レコード変換機能(旧形式)から変更されました。

表2.13 変換エラーファイルの出力

レコード変換機能(旧形式)	レコード変換機能
MDPcvinf構造体のメッセージ出力先ファイル名(msgfname)で指定したメッセージファイルに、変換エラーが発生する度に出力します。	MDPcvinf構造体の変換エラーファイル名(convErrFile)で指定した変換エラーファイルに、すべてのレコード変換処理(MDP_conv)の後、変換エラー出力処理(MDP_errout)を呼び出すことで出力します。

注意

レコード変換機能の変換エラー出力処理(MDP_errout)で出力される変換エラーファイルの形式は、画面操作での実行時と同様です。

詳細は、「ユーザーズガイド 画面操作編」の「3.2.8 変換エラー表示」の「変換エラーファイルの出力内容」を参照してください。

レコード変換機能(旧形式)のメッセージファイルとは異なるので注意してください。

第3章 注意事項

レコード変換機能の注意事項について説明します。

共通の注意事項

- レコード変換機能を使用するには、事前にMDPORTのインストールが必要です。

  また、実行時にはMDPORTのインストール先フォルダへのパスを通してください。

- レコード変換機能では、COBOLファイルのファイル形式を意識しません。
- レコード変換機能では、Unicodeファイルのシグネチャを認識しません。ユーザーアプリケーション側で認識し、レコードデータのみをレコード変換機能に渡してください。

  また、Unicode変換後のシグネチャの付与についても、必要に応じて、ユーザーアプリケーション側で行ってください。

- レコード変換機能で、変換中にシステムエラー等の継続不可能なエラーが発生した場合、当該レコードの変換結果は保証されません。
- レコード変換機能は、マルチスレッドアプリケーションには対応していません。
- レコード変換機能には、文字コード・ファイル形式変換機能(画面操作)でのレコード抽出機能はありませんが、レイアウト定義機能でのマルチレイアウト定義によるレコード抽出は行えます。
- レコード変換機能では、文字コード・ファイル形式変換機能(画面操作)の環境設定で指定された以下の表の項目は参照しません。

  MDPcvinf構造体の指定に従って動作します。

表3.1 文字コード・ファイル形式変換機能(画面操作)の環境設定の項目に対応するMDPcvinf構造体での指定方法と動作

文字コード・ファイル形式変換機能(画面操作)の環境設定の項目	MDPcvinf構造体での指定方法と動作
[基本情報]タブ バッチ実行時のログファイル	MDPcvinf構造体の「logFile」に指定されたファイル名でログファイルを作成します。
[COBOL]タブ BYTE単位の領域長を扱う	MDPcvinf構造体の「binaryByte」の指定に従って、2進項目の領域長をBYTE単位で扱うか判断します。
[COBOL]タブ 一般ファイルをCOBOLの仕様で扱う	MDPcvinf構造体の「inCode」および「outCode」に、COBUB/COBUL/COB16B/COB16L/COB32B/COB32Lを指定した場合、COBOLの仕様で扱います。 COBOLの仕様では、一部の属性について以下のようなコード体系で表されています。 ・ 英数字項目(X-英数字)、混在項目(M-混在):UTF8 ・ 日本語項目(N-日本語):UCS2/UTF16/UTF32 (コード体系に依存) UCS2B/UCS2L/UTF16B/UTF16L/UTF32B/UTF32L/UTF8を指定した場合、COBOLの仕様とはなりません。

MDP_init()関数における注意事項

- レコード変換機能を使用するには、最初に必ずMDP_init()関数を実行してください。
- MDP_init()関数のパラメタとして指定する変換指示構造体(MDPcvinf)は、最初に必ず構造体全体を初期化してください。

MDP_conv()関数における注意事項

- 入出力レコードがレコード区切り文字を含む場合、それぞれのレコード長は、レコード区切り文字長を含むものとします。
- 入出力レコードのデータ長に指定できる最大レコード長は、32767バイトまでです。

・ 入出力レコードが可変長レコードの場合、RDW(Record Descriptor Word)は含まないものとします。

MDP_stat()関数における注意事項

・ MDP_stat()関数では、MDP_init()関数の情報取得はできません。MDP_init()関数の詳細情報は、ログファイルの内容、または標準エラー出力を参照してください。

・ MDP_stat()関数から返される詳細復帰情報は、直前に呼び出されたMDP_conv()関数についてのみの情報です。

対象とするMDP_conv()関数から復帰後、本関数を呼び出す前に他のシステムコールや関数を呼び出すと、詳細復帰情報は無効になります。

MDP_errout()関数における注意事項

・ 変換不能文字がある場合、MDPcvinf構造体の「convErrFile」で指定したファイルへ、変換元コードが16進形式で出力されます。

付録A　関数一覧

レコード変換機能の各関数の仕様について説明します。

表A.1 関数一覧

No	関数名	機能
1	MDP_init()	変換パスの宣言、レイアウト定義ファイルの指定、変換に使用する領域の動的確保等、レコード変換機能の初期処理を行います。
2	MDP_conv()	MDP_init()関数で指示された内容に従って、1レコード分の文字コード・レコード形式変換を行います。
3	MDP_fin()	MDP_init()関数で確保した領域の解放等、レコード変換機能の終了処理を行います。
4	MDP_stat()	直前のMDP_conv()関数での詳細復帰情報を取得します。
5	MDP_errout()	レコード変換時に発生したエラーを収集し、変換エラーファイルへ出力します。

関数の記述形式

関数の記述形式は、以下の規約で記述しています。

- 各関数は、C言語の形式で説明しています。
- 日本語を含まない文字列で記述されている語句は、そのとおりに入力することを示しています。
- 日本語を含む文字列で記述されている語句は、置き換えて入力することを示しています。
- 表中で下線のある語句は、省略時に指定される値を表します。

A.1　MDP_init()

機能概要

変換パスの宣言、レイアウト定義ファイルの指定、変換に使用する領域の動的確保等、レコード変換機能の初期処理を行います。

機能説明

「変換指示構造体形式のパラメタ(cvinf)」のレコード変換指示に従い、文字コード変換テーブルの展開、レイアウト定義情報の展開、エラー情報領域の確保を行います。

レコード変換情報として、その先頭アドレスを復帰値として返します。

形式

```
char *MDP_init(struct MDPcvinf *cvinf)
```

表A.2 パラメタ情報

名称	入力/出力	意味
cvinf	入力	レコード変換情報の作成に必要となる指定値を設定した変換指示構造体(MDPcvinf)

復帰値

本関数の復帰値は、以下のとおりです。

表A.3 復帰値

復帰値	意味
レコード変換情報の先頭アドレス	正常終了の場合は、確保されたレコード変換情報の先頭アドレスが返されます。
NULL	異常終了の場合は、NULLが返されます。

A.1.1 MDPcvinf構造体

レコード変換機能 変換指示構造体です。

形式

```
struct MDPcvinf {               /* MDPORTレコード変換指示構造体          */
       char    inCode[8];       /* 入力文字コード                        */
       char    outCode[8];      /* 出力文字コード                        */
       char    inCodeKey[20];   /* 入力文字コードキーワード              */
       char    outCodeKey[20];  /* 出力文字コードキーワード              */
       int     inRecDlm;        /* 入力レコード改行コード                */
       int     outRecDlm;       /* 出力レコード改行コード                */
       char    outAlt1Char[2];  /* 1バイト系代替コード                   */
       char    outAlt2Char[2];  /* 2バイト系代替コード                   */
       int     errMsgMax;       /* メッセージ出力MAX件数                 */
       int     errMsgNonStop;   /* MSG上限到達時の処理                   */
       int     csvType;         /* CSV参考設定(ローダ形式)               */
       char    csvDlm;          /* CSV区切り1バイト文字                  */
       char    csvQuot;         /* CSV引用符1バイト文字                  */
       int     csvQuotd;        /* CSVデータ中の引用符の扱い             */
       int     csvNoPreBlankCut;/* CSV前置ブランクカット抑制             */
       int     csvNull;         /* CSVヌル項目の引用符の扱い             */
       int     csvNoPsign;      /* CSV＋符号出力抑制                     */
       int     noConv;          /* 同一コード無変換指定                  */
       int     binaryByte;      /* 2進項目をBYTE単位で扱う               */
       int     iconvUse;        /* Charset Managerの使用/不使用          */
       char    *layoutFile;     /* レイアウト定義ファイル名(フルパス)    */
       char    *usrDefTblFile;  /* 利用者定義変換ファイル名(フルパス)    */
       char    *convErrFile;    /* 変換エラーファイル名(フルパス)        */
       char    *logFile;        /* ログファイル名(フルパス)              */
       int     logDsp;          /* ログファイルの内容を標準エラーにも出力 */
       char    *reserve1;       /* 未使用                                */
       char    reserve2[64];    /* 未使用                                */
};
```

メンバ情報

inCode

入力レコードの文字コードを、文字列で指定します。

以下に指定できる文字コードの意味を示します。

表A.4 inCodeの指定内容

指定内容	意味
SJIS	シフトJISコード
EUC	ASCIIおよびEUCコード
JIS	JIS8単位コードおよびJIS漢字コード
JEF	EBCDIC(カナ)およびJEFコード
JEFA	EBCDIC(ASCII)およびJEFコード
Unicode形式	UCS2形式、UTF8形式、UTF16形式、UTF32形式およびCOBOLのUnicode形式
他社コード名	EBCDIC(カナ)および他社日本語コード

Unicode形式には、以下の指定を行うことができます。

表A.5 Unicode形式

Unicode	意味
UCS2B	UCS2(ビッグエンディアン)形式
UCS2L	UCS2(リトルエンディアン)形式
UTF8	UTF8形式
UTF16B	UTF16(ビッグエンディアン)形式
UTF16L	UTF16(リトルエンディアン)形式
UTF32B	UTF32(ビッグエンディアン)形式
UTF32L	UTF32(リトルエンディアン)形式
COBUB	COBOLファイルのUnicode形式 日本語項目はUCS2(ビッグエンディアン)、英数字項目/英数字日本語混在項目はUTF8として扱います。
COBUL	COBOLファイルのUnicode形式 日本語項目はUCS2(リトルエンディアン)、英数字項目/英数字日本語混在項目はUTF8として扱います。
COB16B	COBOLファイルのUnicode形式 日本語項目はUTF16(ビッグエンディアン)、英数字項目/英数字日本語混在項目はUTF8として扱います。
COB16L	COBOLファイルのUnicode形式 日本語項目はUTF16(リトルエンディアン)、英数字項目/英数字日本語混在項目はUTF8として扱います。
COB32B	COBOLファイルのUnicode形式 日本語項目はUTF32(ビッグエンディアン)、英数字項目/英数字日本語混在項目はUTF8として扱います。
COB32L	COBOLファイルのUnicode形式

Unicode	意味
	日本語項目はUTF32(リトルエンディアン)、英数字項目/英数字日本語混在項目はUTF8として扱います。

他社コード名には、以下の指定を行うことができます。

表A.6 他社コード名

他社コード名	意味
IBM	EBCDIC(カナ)およびIBM漢字コード
KEIS	EBCDIC(カナ)および日立KEISコード
JIPE	EBCDIC(カナ)および日本電気JIPS(E)コード
JIPJ	EBCDIC(カナ)および日本電気JIPS(J)コード
AVX	EBCDIC(カナ)および日本電気AVX日本語コード

outCode

出力レコードの文字コードを、文字列で指定します。

指定できる文字コードの意味は、inCodeと同様です。

レコード変換機能で指定できるinCode、outCodeの文字コードの組み合わせについては、「ユーザーズガイド 画面操作編」の「第5章 変換仕様」の「5.1 文字コードの変換」を参照してください。

inCodeKey

Charset Manager変換を行う場合、inCodeで指定した文字コードが、Charset Manager(iconv_open関数)のどのキーワードに該当するかを、文字列で指定します。

文字コードに対しキーワード(指定内容)が一つのキーワードは省略可能です。

指定された文字コード別に以下の指定を行うことができます。

各キーワードの詳細については、Charset Managerのマニュアルを参照してください。

表A.7 inCodeKeyの指定内容

文字コード	指定内容	意味
SJIS	SJIS	シフトJIS(MS/R90/DOS) ※Charset Managerの環境設定に依存
	SJISDOS	シフトJIS(DOS)
	SJISMS	シフトJIS(MS)
EUC	U90	EUC(U90)※1
	S90	EUC(S90)
JIS	JISKANA8	JIS＋8単位半角カナ
JEF	JEFKANA	JEF＋EBCDIC(カナ) (字形重視/領域重視) ※Charset Managerの環境設定に依存
	JEFAUGKANA	JEF＋EBCDIC(カナ) 字形重視
	JEFCOREKANA	JEF＋EBCDIC(カナ) 領域重視
JEFA	JEFASCII	JEF＋EBCDIC(ASCII) (字形重視/領域重視)

文字コード	指定内容	意味
		※Charset Managerの環境設定に依存
	JEFAUGASCII	JEF＋EBCDIC(ASCII) 字形重視
	JEFCOREASCII	JEF＋EBCDIC(ASCII) 領域重視
UCS2B	UCS2	UCS2ビッグエンディアン
UCS2L	UCS2LE	UCS2リトルエンディアン
UTF8	UTF8	UTF8(1～3バイト)
	UTF8_4	UTF8(1～4バイト)
UTF16B	UTF16BE	UTF16ビッグエンディアン
UTF16L	UTF16LE	UTF16リトルエンディアン
UTF32B	UTF32BE	UTF32ビッグエンディアン
UTF32L	UTF32LE	UTF32リトルエンディアン
IBM	I_DBCSKANA	DBCS-Host＋EBCDIC(カナ)
KEIS	H_KEIS	KEIS＋EBCDIC(カナ)
JIPJ	N_JIPSJEAUG	JIPS(J)＋EBCDIC(カナ) 字形重視
	N_JIPSJECORE	JIPS(J)＋EBCDIC(カナ) 領域重視
JIPE	N_JIPSEAUG	JIPS(E)＋EBCDIC(カナ) 字形重視
	N_JIPSECORE	JIPS(E)＋EBCDIC(カナ) 領域重視

備考) 文字コードとしてCOBOLファイルのUnicode形式(COBUB/COBUL/COB16B/COB16L/COB32B/COB32L)を指定した場合は、本キーワードを指定できません。指定した場合は、エラーとなります。

※1 Charset Managerの環境設定で代表コード系キーワード値にs90を指定している場合は、本オペランドにu90を指定することはできません。指定した場合、エラーとなります。

outCodeKey

Charset Manager変換を行う場合、outCodeで指定した文字コードが、Charset Manager(iconv_open関数)のどのキーワードに該当するかを、文字列で指定します。

指定できるキーワードは、inCodeKeyと同様です。

inRecDlm

入力レコードの区切り文字を指定します。

表A.8 inRecDlmの指定内容

指定内容	意味
0 (RECDLM_NONE)	なし
1 (RECDLM_CRLF)	改行コード(CR+LF)
2 (RECDLM_LF)	改行コード(LF)

outRecDlm

出力レコードの区切り文字を指定します。

指定できる区切り文字は、inRecDlmと同様です。

テキスト変換では、出力レコードの区切り文字を指定した場合は可変長形式で、出力レコードの区切り文字を指定しない場合は出力領域のバイト長の固定長形式でレコードを返します。

outAlt1Char

1バイト系コードの変換において、変換不可能な文字が発生した場合、代替文字として出力する文字の文字コードを16進形式で指定します。

出力文字コードがUnicode以外の場合は出力文字コードの1バイトで指定、Unicodeの場合はUCS2(ビッグエンディアン)形式の2バイトで指定します。

省略した場合、“_”(アンダースコア)を代替文字とします。

指定例は、以下のとおりです。

表A.9 代替文字“X”の指定例

出力文字コード	16進数	指定例
SJISの場合	0x58	outAlt1Char[0]=0x58;
Unicodeの場合	0x0058	outAlt1Char[0]=0x00; outAlt1Char[1]=0x58;

outAlt2Char

2バイト系コードの変換において、変換不可能な文字が発生した場合、代替文字として出力する文字の文字コードを16進形式で指定します。

出力文字コードがUnicode以外の場合は出力文字コードの2バイトで指定、Unicodeの場合はUCS2(ビッグエンディアン)形式の2バイトで指定します。

省略した場合、“■”(黒い四角文字)を代替文字とします。

指定例は、以下のとおりです。

表A.10 代替文字“__”(全角アンダースコア)の指定例

出力文字コード	16進数	指定例
SJISの場合	0x8151	outAlt2Char[0]=0x81; outAlt2Char[1]=0x51;
Unicodeの場合	0xFF3F	outAlt2Char[0]=0xFF; outAlt2Char[1]=0x3F;

errMsgMax

変換時に発生した変換エラーの出力件数の上限値を、“0～9999”の数値で指定します。

表A.11 errMsgMaxの指定内容

指定内容	意味
0	デフォルトの“100”が指定されたとみなします。
1～9999	指定された値を、変換エラーの出力件数の上限値とします。

errMsgNonStop

変換時に発生した変換エラーの件数が、errMsgMaxで指定した値に達したとき、その後の変換処理を続行するかどうかを指定します。

表A.12 errMsgNonStopの指定内容

指定内容	意味
0 (ERRMSGNONSTOP_STOP)	処理を中止します。

指定内容	意味
1 (ERRMSGNONSTOP_NO)	エラーを出力せず処理を続行します。
2 (ERRMSGNONSTOP_YES)	エラーを出力して処理を続行します。

csvType

CSV形式レコードを扱う場合に、対象のRDB向けローダ形式(参考設定)を指定します。

レイアウト定義の入力または出力のデータ形式がCSV形式の場合に有効となり、それ以外では無視されます。

なお、レコード変換機能では、項目名ヘッダを先頭レコードに出力はサポートしていません。

参考設定の指定を行った場合、続くcsvDlm、csvQuot、csvQuotd、csvNull、csvNoPsignの指定は無視されます。

これらを有効にしたい場合は、「0 (CSVTYPE_NONE)」を必ず指定して下さい。

参考設定の詳細については、「ユーザーズガイド 画面操作編」の「4.1.6.3.2 CSV情報の指定」の「参考設定について」を参照してください。

表A.13 csvTypeの指定内容

指定内容	意味
0 (CSVTYPE_NONE)	CSV参考設定を使用しない
1 (CSVTYPE_RDB2)	Symfowareのローダ形式
2 (CSVTYPE_ORACLE)	Oracleのローダ形式
3 (CSVTYPE_INFORMIX)	Informixのローダ形式
4 (CSVTYPE_SQLSERVER)	SQL Serverのローダ形式
5 (CSVTYPE_EXCEL)	ExcelのCSV形式

csvDlm

CSV形式の項目間の区切り文字を指定します。

ASCII文字1バイトを指定してください。なお、区切り文字を指定しない場合はNULLを指定します。

入力がCSV形式の場合に、NULLを指定するとカンマが指定されたとみなします。

指定できる区切り文字については、「ユーザーズガイド 画面操作編」の「7.2.2 CSV形式で扱える引用符・区切り文字」を参照してください。

csvQuot

CSV形式の引用符を指定します。

ASCII文字1バイトを指定してください。なお、引用符を指定しない場合はNULLを指定します。

指定できる引用符については、「ユーザーズガイド 画面操作編」の「7.2.2 CSV形式で扱える引用符・区切り文字」を参照してください。

csvQuotd

CSV形式のデータ中に引用符が存在する場合の扱いを指定します。

表A.14 csvQuotdの指定内容

指定内容	意味
0 (QUOTD_OFF)	引用符を2つ並べない
1 (QUOTD_ON)	引用符を2つ並べる

csvNoPreBlankCut

CSV形式の入力データが引用符で括られていないときに、前置の空白文字の削除を抑制するかどうかを指定します。

表A.15 csvNoPreBlankCutの指定内容

指定内容	意味
0 (CSVNOPREBLANKCUT_OFF)	前置ブランクカットを抑制しない
1 (CSVNOPREBLANKCUT_ON)	前置ブランクカットを抑制する

csvNull

出力のCSV形式のデータ中にNULL文字列を、引用符で括って出力するかどうかを指定します。

表A.16 csvNullの指定内容

指定内容	意味
0 (CSVNULL_OFF)	引用符で括らない
1 (CSVNULL_ON)	引用符で括る

csvNoPsign

出力のCSV形式の数値文字列データに、＋符号の出力を抑制するかどうかを指定します。

表A.17 csvNoPsignの指定内容

指定内容	意味
0 (NOPSIGN_OFF)	抑制しない(＋符号を出力する)
1 (NOPSIGN_ON)	抑制する(＋符号を出力しない)

noConv

inCodeおよびoutCodeが同一文字コードの場合に、無変換出力を行うかどうかを指定します。

表A.18 noConvの指定内容

指定内容	意味
0 (NOCONV_OFF)	無変換出力を行わない(変換処理を行う)
1 (NOCONV_ON)	無変換出力を行う(変換処理を行わない)

binaryByte

2進項目の領域長を、バイト単位で扱うかどうかを指定します。

表A.19 binaryByteの指定内容

指定内容	意味
0 (BINBYTE_OFF)	BYTE単位で扱わない(WORD単位で扱う)
1 (BINBYTE_ON)	BYTE単位で扱う

iconvUse

Charset Managerの使用有無を指定します。

表A.20 iconvUseの指定内容

指定内容	意味
1 (ICONVUSE_ON)	Charset Managerを使用します。 Charset Managerがインストールされていない場合や、Charset Managerで対応しない文字コード変換パスの場合には、コードテーブル作成エラーを発生し、処理を中止します。
2 (ICONVUSE_OFF)	Charset Managerを使用しません。 Charset Managerが必須の文字コード変換パスの場合には、コードテーブル作成エラーを発生し、処理を中止します。

layoutFile

レイアウト定義ファイル名を、フルパスで指定します。

テキスト変換以外の場合は、必ず指定してください。

テキスト変換の場合は、NULLを指定してください。

指定されたファイル名が存在しない場合は、異常終了します。

usrDefTblFile

利用者定義変換テーブルのファイル名を、フルパスで指定します。

利用者定義変換テーブルを使用しない場合は、NULLを指定してください。

指定されたファイル名が存在しない場合は、異常終了します。

利用者定義変換テーブルに関する詳細は、「ユーザーズガイド 画面操作編」の「付録A 利用者定義変換テーブル」を参照してください。

convErrFile

MDP_errout()関数で出力する変換エラーファイルの出力先ファイル名を、フルパスで指定します。

変換エラーファイルを出力しない場合は、NULLを指定してください。

指定されたファイル名が存在しない場合は、新規に作成します。

指定されたファイル名が既に存在する場合は、MDP_errout()関数の出力モードmodeにより上書き、または追加書きします。

logFile

ログファイル名を、フルパスで指定します。

ログファイル名は必ず指定する必要があります。

指定しない場合は、異常終了します。

指定されたファイル名が存在しない場合は、新規に作成します。

指定されたファイル名が既に存在する場合は、追加書きします。

logDsp

ログファイルに出力する内容を、標準エラーにも出力するかどうかを指定します。

表A.21 logDspの指定内容

指定内容	意味
0 (LOGDSP_OFF)	標準エラーに出力しない
1 (LOGDSP_ON)	標準エラーにも出力する

reserve1

使用しません。

reserve2

使用しません。

A.2 MDP_conv()

機能概要

MDP_init()関数で指示された内容に従って、1レコード分の文字コード・レコード形式変換を行います。

機能説明

MDP_init()関数で取得したレコード変換情報(inf)に従い、入力レコード(irec)内の入力レコード長(ireclen)分を変換対象として、文字コードの変換・レコード形式の変換を行います。

変換結果を出力レコード(orec)に、変換後有効バイト長を出力レコード長(oreclen)に、それぞれ格納します。

形式

```
int MDP_conv(char *inf, char *irec, int ireclen, char *orec, int *oreclen)
```

表A.22 パラメタ情報

名称	入力/出力	意味
inf	入力	MDP_init()関数から返却されたレコード変換情報の先頭アドレス
irec	入力	入力レコード格納領域の先頭アドレス
ireclen	入力	入力レコードの有効データバイト長
orec	出力	出力レコード格納領域の先頭アドレス
oreclen	入力/出力	入力:確保した出力レコード格納領域の領域バイト長 出力:出力(変換結果)レコードの有効データバイト長

備考) orecの領域は、ireclen、文字コード、レコードタイプ等を考慮したうえで、呼び出し元で十分な領域を確保し、その領域バイト長をoreclenに設定してください。

復帰値

本関数の復帰値は、以下のとおりです。

表A.23 復帰値

復帰値	意味
0 (MDPORT_OK)	正常終了
-1 (MDPORT_NG)	異常終了

復帰値	意味
-2 (MDPORT_SKIP)	出力対象外のレコード(レイアウト定義において、マルチレイアウト等での、レイアウトの適用条件に該当しないレコード)

A.3 MDP_fin()

機能概要

MDP_init()関数で確保した領域の解放等、レコード変換機能の終了処理を行います。

機能説明

MDP_init()関数で取得したレコード変換情報(inf)の領域を解放し、レコード変換の終了処理を行います。

形式

```
int MDP_fin(char *inf)
```

表A.24 パラメタ情報

名称	入力/出力	意味
inf	入力	MDP_init()関数から返却されたレコード変換情報の先頭アドレス

復帰値

本関数の復帰値は、以下のとおりです。

表A.25 復帰値

復帰値	意味
0 (MDPORT_OK)	正常終了
-1 (MDPORT_NG)	異常終了

A.4 MDP_stat()

機能概要

直前のMDP_conv()関数での詳細復帰情報を取得します。

機能説明

直前のMDP_conv()関数での処理結果を、取得情報の指定(para)に従った情報で取得します。

形式

```
int MDP_stat(int para, char *inf)
```

表A.26 パラメタ情報

名称	入力/出力	意味
para	入力	取得情報の指定
inf	入力	MDP_init()関数から返却されたレコード変換情報の先頭アドレス

表A.27 paraの指定内容

指定内容	意味
1 (MDP_WARNUM)	変換エラー発生件数を取得します。
2 (MDP_ERROR)	直前のMDP_conv()関数で発生した詳細復帰コードを取得します。

復帰値

paraに指定した値により、以下の値が返されます。

表A.28 “MDP_WARNUM”の場合の復帰値

復帰値	意味
0	変換エラーなし
正の整数	変換エラー発生件数

表A.29 “MDP_ERROR”の場合の復帰値

復帰値	意味
0	正常終了しました。
-1 (MDPERR_PARA)	引数に誤りがあります。
-2 (MDPERR_CONV)	変換エラーが発生しました。
-3 (MDPERR_LLNG)	入出力レコード長に誤りがあります。
-4 (MDPERR_ERROVER)	変換エラーの件数が上限値を超えました。
-5 (MDPERR_TRYUSE)	体験版で扱える最大レコード数を超えました。
その他の数値	システムのerrnoです。

A.5 MDP_errout()

機能概要

レコード変換時に発生したエラーを収集し、変換エラーファイルへ出力します。

機能説明

変換時に発生したエラーを収集し、変換エラーファイルへ出力モード(mode)に従って出力します。

なお、本関数で変換エラーファイルを出力した場合、変換エラーファイル中のカレントフォルダおよび入力ファイル名は、空白で出力されます。

変換エラーファイルの出力形式の詳細については、「ユーザーズガイド 画面操作編」の「第3章 操作説明」の「3.2.8 変換エラー表示」の「変換エラーファイルの出力例」および「変換エラーファイルの出力内容」を参照してください。

形式

```
int MDP_errout(char *inf, int mode)
```

表A.30 パラメタ情報

名称	入力/出力	意味
inf	入力	MDP_init()関数から返却されたレコード変換情報の先頭アドレス
mode	入力	出力モード

表A.31 modeの指定内容

指定内容	意味
0	指定した変換エラーファイルが既に存在する場合は、上書きします。
1	指定した変換エラーファイルが既に存在する場合は、追加書きします。

復帰値

本関数の復帰値は、以下のとおりです。

表A.32 復帰値

復帰値	意味
0 (MDPORT_OK)	正常終了
-1 (MDPORT_NG)	異常終了

付録B　使用例

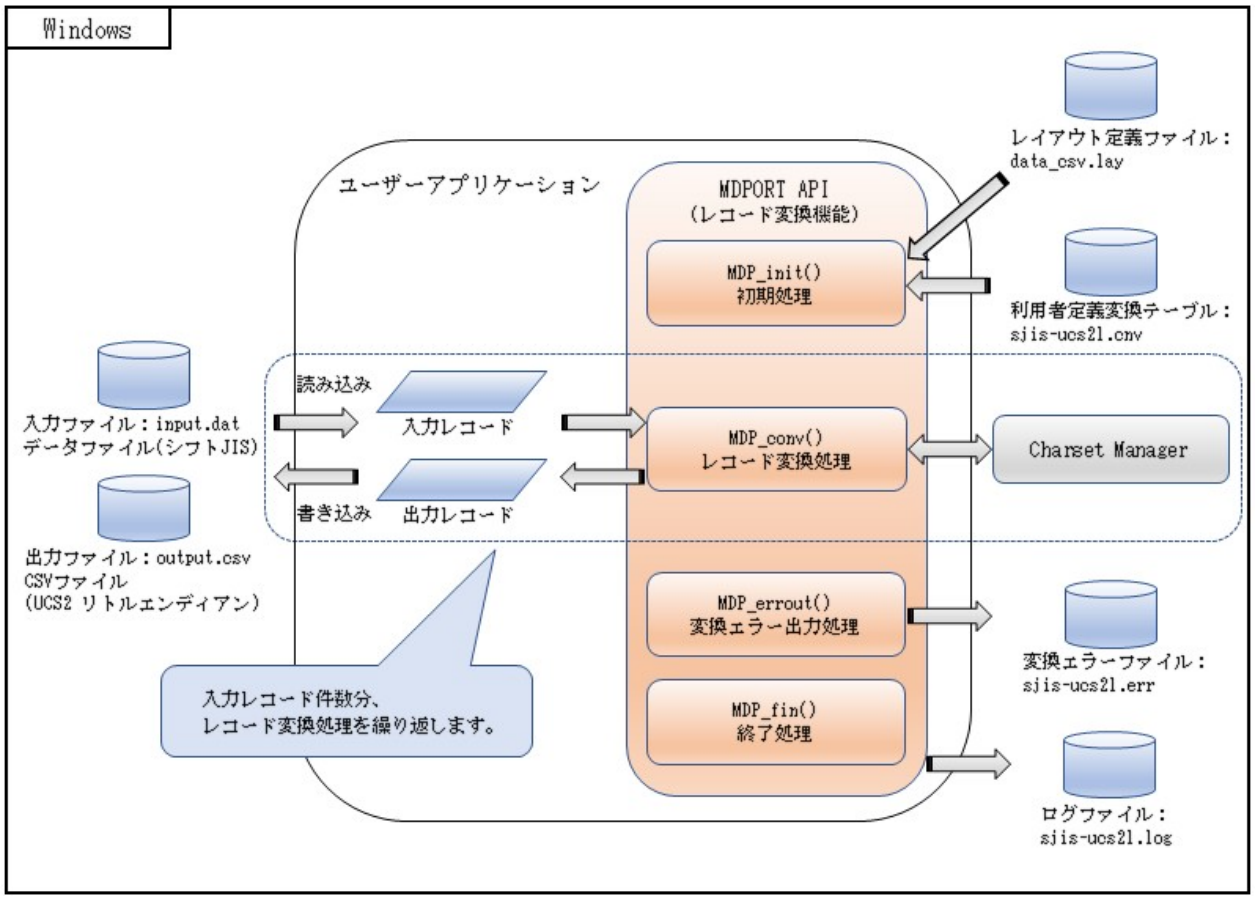
MDPORTのAPIの使用例について説明します。

要件

Windows上の固定長ファイル(シフトJIS)を、CSVファイル(UCS2 リトルエンディアン)へ変換します。

C言語プログラムからレコード変換機能の関数を使用します。

図B.1 変換のイメージ図



ソースコードの記述例 (C言語プログラム)

```
#include <io.h>
#include <fcntl.h>
#include <string.h>
#include <sys/stat.h>

#include "MDRcomm.h"

int main()
{
    struct MDPcvinf cvinf;
    char *mdpinf;
    char indata[500];
    char outdata[1000]; /*レコード長の増加を考慮し、余裕を持たせた出力領域を確保 */
```

```
int  inlen;
int  outlen;
int  rtncod;
int  ifd;
int  ofd;
char *input_file = "D:¥¥data¥¥input.dat";
char *output_file = "D:¥¥data¥¥output.csv";
int  readlen = 500;


/*****************************************
* データ(SJIS)→CSV(UCS2L)の初期処理
*****************************************/
memset(&cvinf, 0x00, sizeof(struct MDPcvinf));
strcpy(cvinf.inCode, "SJIS");                   /* 入力文字コード : SJIS                    */
strcpy(cvinf.outCode, "UCS2L");                 /* 出力文字コード : UCS2L                   */
strcpy(cvinf.inCodeKey, "SJISMS");              /* 入力文字コードキーワード : SJISMS        */
strcpy(cvinf.outCodeKey, "UCS2LE");             /* 出力文字コードキーワード : UCS2LE        */
cvinf.inRecDlm = RECDLM_NONE;                   /* 入力レコード改行コード : 改行なし        */
cvinf.outRecDlm = RECDLM_CRLF;                  /* 出力レコード改行コード : CRLF            */
cvinf.outAlt1Char[0] = 0x00;                    /* 1バイト系代替コード 1バイト目            */
cvinf.outAlt1Char[1] = 0x78;                    /* 1バイト系代替コード 2バイト目            */
cvinf.outAlt2Char[0] = 0xFF;                    /* 2バイト系代替コード 1バイト目            */
cvinf.outAlt2Char[1] = 0x3F;                    /* 2バイト系代替コード 2バイト目            */
cvinf.errMsgMax = 9999;                         /* メッセージ出力MAX件数 : 9999             */
cvinf.errMsgNonStop = ERRMSGNONSTOP_STOP;       /* MSG上限到達時の処理 : 処理ストップ       */
cvinf.csvType = CSVTYPE_NONE;                   /* CSV参考設定(ローダ形式) : 指定しない     */
cvinf.csvDlm = ',';                             /* CSV区切り1バイト文字 : カンマ            */
cvinf.csvQuot = '"';                            /* CSV引用符1バイト文字 : ダブルクォート    */
cvinf.csvQuotd = QUOTD_ON;                      /* CSVデータ中の引用符の扱い : 2つ並べる    */
cvinf.csvNull = CSVNULL_ON;                     /* CSVヌル項目の引用符の扱い : 引用符で括る */
cvinf.csvNoPsign = NOPSIGN_ON;                  /* CSV＋符号出力抑制 : ＋符号を出力しない   */
cvinf.iconvUse = ICONVUSE_ON;                   /* Charset Managerの使用/不使用 : 使用する  */
cvinf.layoutFile = "D:¥¥lay¥¥data_csv.lay";     /* レイアウト定義ファイル名(フルパス)       */
cvinf.usrDefTblFile = "D:¥¥usr¥¥sjis-ucs2l.cnv"; /* 利用者定義変換ファイル名(フルパス)      */
cvinf.convErrFile = "D:¥¥err¥¥sjis-ucs2l.err";  /* 変換エラーファイル名(フルパス)           */
cvinf.logFile = "D:¥¥log¥¥sjis-ucs2l.log";      /* ログファイル名(フルパス)                 */
cvinf.logDsp = LOGDSP_OFF;                      /* ログファイルの内容を標準エラーに出力しない */

/* レコード変換初期処理関数呼出し */
mdpinf = MDP_init( &cvinf );


/*****************************************
* データ(SJIS)→CSV(UCS2L)の変換処理
*****************************************/
/* 入力ファイルオープン */
ifd = open(input_file, O_RDONLY);

/* 出力ファイルオープン */
ofd = open(output_file, O_CREAT | O_WRONLY | O_TRUNC | O_BINARY, S_IREAD | S_IWRITE);

/* 入力レコード繰り返し */
while (1)
{
    /* 初期化 */
    memset(&indata[0], 0x00, sizeof(indata));
    memset(&outdata[0], 0x00, sizeof(outdata));
    outlen = sizeof(outdata);

    /* 入力ファイル読み込み */
    inlen = read(ifd, indata, readlen);
    if (inlen <= 0)
```

```
        {
            break;
        }


        /* レコード変換関数呼び出し */
        rtncod = MDP_conv( mdpinf, indata, inlen, outdata, &outlen );

        /* 出力ファイル書き込み */
        write(ofd, outdata, outlen);
    }

    /* 入出力ファイルクローズ */
    close(ifd);
    close(ofd);


    /*****************************************
    * 変換エラー出力処理
    *****************************************/
    rtncod = MDP_errout( mdpinf, 0 );


    /*****************************************
    * レコード変換後処理
    *****************************************/
    rtncod = MDP_fin( mdpinf );


    return 0;
}
```

用語集

Charset Manager

富士通標準コード変換を提供しているWindows上のソフトウェアです。

動作保証しているCharset Managerのバージョン・レベルについては、MDPORTの「ソフトウェア説明書」を参照してください。

Charset Manager変換

Charset Managerを使用した変換を、MDPORTではCharset Manager変換と呼んでいます。

詳しくは、「ユーザーズガイド 画面操作編」の「5.1.3 2バイト系コード変換」を参照してください。

COBOLファイル

Windows上のCOBOLアプリケーションプログラムでアクセスするファイル形式です。

ファイル編成の種類として、レコード順ファイル、行順ファイル、相対ファイル、索引ファイルがあります。

CSV形式

項目間をカンマ(区切り文字)で区切った形式のテキストレコードを指します。

MDPORTでは区切り文字がカンマ以外でもCSV形式と呼びます。一般的に可変長レコードです。

CSV形式であれば、表計算ソフトや各種RDBローダファイル、awk等のユーティリティへと応用することができます。

Informix

IBM社のリレーショナルデータベース管理システム(RDBMS)です。

IBM社が買収した企業(Informix社)が開発したことが起源となっています。

Oracle

オラクル社が開発したリレーショナルデータベース管理システム(RDBMS)です。

SQL Server

マイクロソフト社が開発したリレーショナルデータベース管理システム(RDBMS)です。

Symfoware

富士通が開発したリレーショナルデータベース管理システム(RDBMS)です。

シグネチャ

Unicodeのエンディアンを示すためにファイルの先頭に付加される数バイトのデータです。

シグネチャにより、UCS2/UTF16(リトルエンディアン)、UCS2/UTF16(ビッグエンディアン)、UTF32(リトルエンディアン)、UTF32(ビッグエンディアン)、UTF8のどれなのかを識別することができます。

BOM(Byte Order Mark)と同じ意味です。

シフトコード

1バイト文字と2バイト文字の切替えのための制御コードのことです。

データレコード変換

複数の項目で構成されたレイアウトを意識したレコード変換モードです。

一般的にアプリケーションでの処理対象となるデータレコードを変換する場合に指定します。

CSV形式レコードを対象とする場合も、データレコード変換を使用します。

テキスト変換

ソース等のテキストレコードを変換するレコード変換モードです。

ビッグエンディアン

低アドレスのバイト位置が上位桁を表す形式です。

SPARC等のUNIXマシンではビッグエンディアンです。

マルチレイアウト

1つのファイル内において、複数のレコードレイアウトが存在することを指します。

MDPORTでは、1つのレイアウト定義ファイルに対して、複数のレイアウトの適用条件を設定することにより、マルチレイアウトのデータを変換することができます。

リトルエンディアン

低アドレスのバイト位置が下位桁を表す形式です。

Windows上のプログラムはリトルエンディアンです。

レイアウト定義ファイル

レイアウト定義機能で作成するファイルです。

データレコード変換を行うには、レイアウト定義ファイルが必須となります。

拡張子は“.lay”です。

字形重視

字形を重視した変換です。

文字がJIS領域にあるか拡張域にあるかには関わらず、新字体は新字体へ、旧字体は旧字体へ、同じ字体のコードの変換を行います。

代替文字

コード変換エラー等で、変換先のコードに変換できなかった文字の代わりに出力される文字です。

標準変換

Charset Managerを使用せずに、MDPORT内部の算術式に従ってコード変換を行う方法を、MDPORTでは標準変換と呼んでいます。

詳しくは、「ユーザーズガイド 画面操作編」の「5.1.3 2バイト系コード変換」を参照してください。

変換エラーファイル

変換エラーが発生した場合に、エラー情報が出力されるファイルです。

テキストファイルの形式で、Windows標準のメモ帳等で内容を参照できます。

拡張子は“.err”です。

利用者定義変換テーブル

利用者定義文字(外字)や拡張漢字/非漢字を任意の文字へ変換するための、コードの対応づけを行うテーブルファイルです。

拡張子は“.cnv”です。

領域重視

JIS領域を重視した変換です。

字体が新字体であるか旧字体であるかには関わらず、JIS領域の文字同士の変換を行います。